Sistem Navigasi *Indoor* Menggunakan *Bi-Directional Dijkstra Search* Berbasis Integrasi dengan *Smartphone* untuk Studi Kasus pada Gedung Bertingkat

Mohammad Ardhiansyah Metana Putra, R. V. Hari Ginardi, dan Abdul Munif
Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember (ITS)
Jl. Arief Rahman Hakim, Surabaya 60111 Indonesia
e-mail: hari@its.ac.id, munif@if.its.ac.id, metana.p12@mhs.if.its.ac.id

***Abstrak*—Saat ini kebutuhan masyarakat akan informasi lokasi sangat tinggi, terutama dengan memanfaatkan teknologi teknologi *Global Positioning System* (GPS). GPS merupakan salah satu teknologi yang bisa digunakan untuk melakukan navigasi diluar ruangan, namun ketika berada di dalam ruangan atau bangunan sistem ini memiliki akurasi yang rendah, apalagi untuk ruangan/gedung yang besar. Oleh karena itu, sebuah sistem yang lebih akurat untuk memberikan solusi bagi pendeteksian lokasi di dalam ruangan atau gedung yang memiliki lebih dari satu level lantai dikembangkan dengan konsep *3D Indoor Navigation System*. Sistem yang dibangun dapat menunjukan posisi pengguna dengan menggunakan *Indoor Localization System* dengan memanfaatkan *Received Signal Strength* (RSS) dan rute navigasi dalam tampilan peta 3D menggunakan *Bi-Directional Algorithm* untuk melakukan pencarian rute terdekat antar dua tempat. Untuk mengembangkan tampilan peta 3D digunakan Unity3D. Hasil pengujian sistem menunjukan bahwa sistem ini dapat menampilkan lokasi pengguna dan rute perjalanan pada peta 3D serta pengguna bisa melakukan navigasi pada peta tersebut. Dapat ditarik kesimpulan bahwa sistem ini dapat lebih membantu pengguna dalam menemukan tempat yang ingin dituju pada studi kasus gedung Teknik Informatika ITS.**

***Kata Kunci*—*Bi-Directional Dijkstra, Indoor Localization, Location Based Service*, Navigasi.**

I. PENDAHULUAN

SAAT ini ada banyak bangunan tinggi dan besar yang terdiri dari berbagai lantai dan ruangan di dalamnya. Pada umumnya gedung-gedung bertingkat menyediakan sistem layanan informasi yang dapat dimanfaatkan oleh pengunjung untuk mendapatkan informasi dan lokasi tujuan. Sistem layanan informasi yang diberikan biasanya berbentuk pos-pos layanan informasi yang dijaga oleh petugas. Pengunjung yang ingin mendapatkan informasi harus bertanya kepada petugas. Namun pada kenyataannya terdapat beberapa permasalahan seperti dibutuhkan waktu yang relatif lama dalam mencari informasi yang dibutuhkan, dan juga informasi yang diberikan petugas belum tentu akurat dan kurang detail. Sehingga membuat pengunjung kesulitan dalam mencari informasi yang dibutuhkan.

Teknologi yang biasa digunakan untuk mendapatkan informasi lokasi dan rute perjalanan adalah *Global Positioning System* (GPS). GPS mempunyai manfaat yang cukup besar, namun sistem ini memiliki akurasi yang rendah saat berada di suatu ruangan atau bangunan. GPS hanya dapat menunjukan daerah atau nama jalan pada lokasi pengguna, akan tetapi sistem ini tidak dapat menunjukan lokasi ruangan pada suatu bangunan dimana pengguna berada dan GPS tidak bisa memberikan rute perjalanan kepada pengguna. Hal ini terjadi karena ketidakmampuan gelombang radio yang dikirimkan oleh satelit untuk menembus benda padat dan tebal seperti tembok ruangan, gedung bertingkat dan lain sebagainya [1].

Seiring dengan kebutuhan informasi yang cepat dalam mendapatkan posisi dan rute perjalanan maka dibutuhkan kekuatan sinyal *Wi-Fi* dan algoritma pencarian rute terpendek. Setiap lokasi yang ada akan memiliki beragam kekuatan sinyal dari *Wi-Fi* yang berbeda, sehingga setiap lokasi akan unik karena kekuatan sinyal *Wi-Fi* akan berbeda. Algortima pencarian rute terpendek menggunakan *vertex*, hal ini digunakan karena algoritma bekerja menggunakan *graph* untuk penentuan rute lintasan terpendek. Untuk mendapatkan navigasi yang akurat, terdapat dua tahap. Tahap pertama, tahap pembuatan peta, pada tahap ini peta dibuat dengan menempatkan *vertex* di tempat-tempat tertentu seperti persimpangan dan di depan ruangan. Tahap kedua, tahap pencarian rute, mencari jarak terpendek dengan *vertex* terdekat yang telah dibuat di peta dan meneruskan rute dengan melanjutkan ke *vertex* berikutnya sampai ke tujuan.

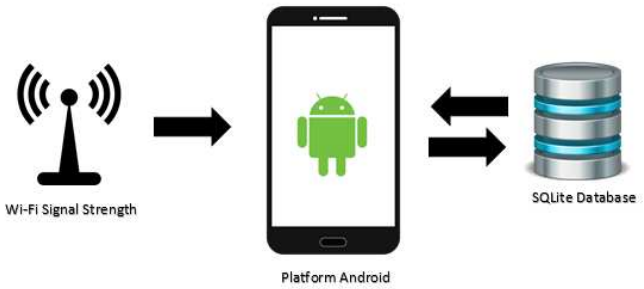
Pada Tugas Akhir ini, *3D Indoor Navigation* akan menggunakan *Indoor Localization* untuk mendapatkan posisi dan *Bi-Directional Dijkstra Search* untuk mencari rute terpendek di Gedung Teknik Informatika ITS.

II. TINJAUAN PUSTAKA

A. Location Based Service (LBS)

Location Based Services (LBS) atau layanan berbasis lokasi adalah layanan informasi yang dapat diakses melalui *mobile device* dengan menggunakan *mobile network*, yang dilengkapi kemampuan untuk memanfaatkan lokasi dari *mobile device* tersebut. Layanan berbasis lokasi dapat digambarkan sebagai suatu layanan yang berada pada pertemuan tiga teknologi yaitu: *Geographic Information System*, *Internet Service*, dan *Mobile*

Device [2]. Ada beberapa teknologi yang digunakan LBS untuk mendapatkan lokasi geografis dari sebuah perangkat *mobile*, di antaranya *Global Positioning System*, *Indoor Positioning System*, dan *Indoor Localization*. Salah satu contoh dari aplikasi yang menggunakan konsep LBS adalah aplikasi *maps*.



Gambar 1. Arsitektur Sistem

B. Indoor Localization Menggunakan Sinyal Wi-Fi dan Clustering Filtered K-Nearest Neighbors

Sistem *indoor localization* yang memiliki kemampuan untuk mendeteksi lokasi pengguna di dalam ruangan dengan menggunakan data kekuatan sinyal *Wi-Fi* yang ditangkap oleh *smartphone* pada ruangan tersebut. Hasil yang didapatkan adalah informasi lokasi pengguna seperti: nama ruangan atau area lokasi keberadaan pengguna serta tingkat lantai dari lokasi tersebut.

Sistem ini telah dikembangkan pada studi kasus pada Gedung Teknik Informatika ITS. Sistem memberikan performa yang baik dengan persentase rata-rata akurasi pendeteksian lokasi sebesar 93,21% untuk seluruh *test area* pada setiap lokasi uji coba [3].

C. Unity3D Game Engine

Unity3D adalah sebuah *game engine* yang dikembangkan oleh Unity Technologies yang juga merupakan *tool* data yang digunakan tidak hanya untuk membuat *game* tetapi juga konten, animasi 3D waktu nyata, dll. Unity dapat berjalan pada Windows dan Macintosh OS X dan digunakan untuk pengembangan pada platform yang bermacam-macam termasuk Windows, Mac, Wii, iPhone, Android, dan lain-lain. Unity juga dapat digunakan untuk membuat game 3D berbasis web dengan menggunakan *plug-in* Unity Web Player [4].

D. SQLite

SQLite adalah sebuah *open source database* sangat terkenal pada perangkat kecil seperti Android karena cukup stabil. Pada *platform* Android, SQLite dijadikan satu di dalam Android *runtime*, sehingga setiap aplikasi Android dapat membuat basis data SQLite. SQLite menggunakan antarmuka SQL, sehingga cukup mudah untuk digunakan [5].

```
1.   set path[] to ""
2.   For each vertex in vertices:
3.      If vertex is not source
4.         Set dist[vertex] to infinity
5.      Else if vertex is source:
6.         Set dist[vertex] to 0
7.      End if
8.      Set nodes to vertex
9.   End for
10.
11.  While nodes is not null:
12.     Set smallest to vertex in nodes with smallest
     distance in dIst[]
13.     Remove smallest from nodes
14.
15.     If smallest is finish:
16.        Add prev[] to path[]
17.        Break
18.     End if
19.     If dist[smallest] is infinity:
20.        Break
21.     End if
22.
23.     For each neighbour in vertices[smallest]:
24.        Set alt = dist[smallest] + distance neighbour
25.
26.        If alt < dist[neighbour]:
27.           Dist[neighbour] = alt
28.           Prev[neighbour] = smallest
29.        End if
30.     End for
31.  End while
```

Kode Sumber 1. *Pseudocode* algoritma *Dijkstra*

E. Algoritma Dijkstra

Algoritma *Dijkstra* merupakan algoritma untuk menemukan jalur terpendek antar *vertex* dalam sebuah *graph*. Algoritma ini dinamakan sesuai dengan penemunya Edsger W. Dijkstra. Edsger Wybe Dijkstra merupakan seorang ilmuwan komputer berkebangsaan Belanda. Algoritma ini ditemukan dipublikasikan pada tahun 1959 [6]. Dalam pengaplikasian algoritma *Dijkstra* dalam suatu permasalahan, diperlukan beberapa syarat berikut:

1. *Graph* yang digunakan dapat berupa *graph* yang memiliki arah maupun yang tidak memiliki arah.
2. Semua *edge* pada *graph* harus memiliki *weight non-negative*.
3. *Graph* harus terhubung.

Pseudocode algoritma *Dijkstra* ditunjukan pada Kode Sumber 1.

F. Algoritma Bi-Directional Dijkstra

Langkah awal untuk menggunakan algoritma *Bi-Directional Dijkstra* adalah menentukan *vertex* awal dan *vertex* akhir. Setelah itu di *vertex* titik tersebut akan dijalankan secara bersamaan menggunakan algoritma *Dijkstra*. Dengan cara ini rute terpendek dapat diperoleh dari data yang telah dikumpulkan. Teknik ini memungkinkan untuk mempercepat pencarian rute dibandingkan algoritma *Dijkstra* biasa [7].

Pseudocode algoritma *Bi-Directional Dijkstra* ditunujukan pada Kode Sumber 2.

```
32.  set listPath[] to ""
33.     For each vertex in vertices:
34.        If vertex is not source and vertex is not
     destination
```

```
35.           Set forwDist[vertex] and backDist[vertex]
     to infinity
36.        Else if vertex is source:
37.           Set forwDist[vertex] to 0
38.           Set backDist[vertex] to infinity
39.        Else if vertex is destination:
40.           Set forwDist[vertex] to infinity
41.           Set backDist[vertex] to 0
42.        End if
43.        Set forwNodes and backNodes to vertex
44.     End for
45.
46.     If source is destination
47.        add source to listPath
48.        Break
49.     End if
50.   Set forwSmallest to vertex in forwNodes with
     smallest distance in forwDIst[]
51.      Set backSmallest to vertex in backNodes with
     smallest distance in backDist[]
52.   While forwNodes and backNodes is not null
53.        // Check
54.        If forwSmallest in backPrev[]
55.           Add forwPrev[] and backPrev[] to listPath[]
56.        Else if backSmallest in forwPrev[]
57.           Add forwPrev[] and backPrev[] to listPath[]
58.        End if
59.        // End Check
60.
61.        Set forwSmallest to vertex in forwNodes with
     smallest distance in forwDIst[]
62.        Remove forwSmallest from forwNodes
63.
64.        For each neighbour in vertices[forwSmallest]
65.           Set alt = forwDist[forwSmallest] + distance
     neighbour
66.           If alt < forwDist[neighbour]
67.              Set forwDist[neighbour] = alt
68.              Set forwPrev[neighbour] = forwSmallest
69.           End if
70.        End for
71.
72.        // Check Again
73.
74.        Set backSmallest to vertex in backNodes with
     smallest distance in backDist[]
75.        Remove backSmallest from backNodes
76.
77.     For each neighbour in vertices[backSmallest]
78.           Set alt = backDist[backSmallest] + distance
     neighbour
79.           If alt < backDist[neighbour]
80.              Set backDist[neighbour] = alt
81.              Set backPrev[neighbour] = backSmallest
82.           End if
83.        End for
84.   End While
```

Kode Sumber 2. *Pseudocode* algoritma *Bi-Directional Dijkstra*

III. ANALISIS DAN PERANCANGAN

A. *Analisis Sistem*

Aplikasi yang akan dibangun merupakan aplikasi berbasis perangkat bergerak Android. Aplikasi ini memiliki kemampuan untuk menentukan posisi, rute terbaik, dan arah perjalan untuk mencapai lokasi tujuan pengguna. Semua informasi tersebut akan disajikan pada layar *smartphone* pengguna.

Kebutuhan utama dalam aplikasi ini antara lain:

1. pengguna dapat mengetahui posisi keberadaannya,
2. pengguna dapat melakukan navigasi

B. *Perancangan Sistem*

Arsitektur sistem pada aplikasi *indoor navigation* ini ditunjukkan pada Gambar 1. Berdasarkan gambar tersebut, informasi lokasi pengguna dapat diketahui dengan mendeteksi data sinyal *Wi-Fi* yang ditangkap pada saat itu kemudian data tersebut diolah untuk mendapatkan informasi lokasi pengguna yang kemudian digunakan sebagai titik awal pada pencarian rute menuju tujuan.

C. *Perancangan Proses dan Alur Sistem*

Proses yang dibangun pada sistem ini meliputi proses prediksi lokasi pengguna, proses pencarian rute terpendek, serta proses menampilkan rute pada peta. Pada proses prediksi lokasi pengguna, data-data mengenai sinyal *Wi-Fi* disekitar pengguna akan diambil dan diolah dengan memanfaatkan sistem *indoor localization*. Setelah sistem *indoor localization* mendapatkan lokasi pengguna maka akan dilakukan proses pencarian rute terpendek dengan menggunakan algoritma *Bi-Directional Dijkstra* data prediksi lokasi pengguna akan digunakan sebagai *data source* dan *data destination* akan dimasukkan manual oleh pengguna. Setelah rute pencarian didapatkan maka rute akan ditampilkan pada peta 3D.

IV. IMPLEMENTASI SISTEM

Pada bagian implementasi sistem, dibangun aplikasi *navigation indoor* berbasis perangkat bergerak Android. Aplikasi ini dibangun dengan bahasa pemrograman C# dengan menggunakan Unity3D *Game Engine* sebagai IDE. Aplikasi ini digunakan untuk menjalankan berbagai fungsionalitas sistem yang ditampilkan dalam antarmuka halaman utama.

Ketika membuka aplikasi untuk pertama kali, sistem akan mengaktifkan koneksi *Wi-Fi* jika sebelumnya belum diaktifkan. Koneksi *Wi-Fi* digunakan untuk mengambil data-data sinyal *Wi-Fi* di sekitar pengguna. Antarmuka halaman utama adalah antarmuka yang berisi semua fitur aplikasi yaitu fitur untuk melakukan prediksi lokasi pengguna saat ini yang ditunjukan dengan tombol berwarna oranye, dan informasi lokasi akan ditunjukan dengan perubahan posisi sudut pandang dan informasi ruangan ditunjukan pada kolom bagian atas, fitur selanjutnya adalah fitur pencarian dan menampilan rute yang ditunjukan dengan dua kotak tulisan yang bisa diisi nama ruangan oleh pengguna sebagai awal dan tujuan navigasi serta tombol panah digunakan untuk mencari dan menampilkan rute pencarian pada peta 3D. Antarmuka halaman utama ditunjukan oleh Gambar 2.

Gambar 2**.** Antarmuka halaman utama

V. PENGUJIAN TERHADAP PENGGUNA

Pengujian terhadap sistem yang telah selesai dibangun meliputi pengujian fungsionalitas pengujian kinerja serta analisa algoritma. Pengujian fungsionalitas dilakukan dengan metode *blackbox*. Pengujian fungsionalitas dikatakan berhasil ketika kondisi akhir sesuai dengan kondisi yang diharapkan. Pengujian fungsionalitas yang dilakukan ada 3 macam yaitu:

1. Menampilkan posisi pengguna, dengan posisi awal dalam gedung belum diketahui kemudian pengguna menekan tombol *GetPosition* pada aplikasi untuk mengetahui lokasi dan mendapatkan informasi pada peta 3D.
2. Menampilkan rute navigasi, pada kondisi sebelumnya tidak ada rute navigasi yang ditampilkan sehingga pengguna perlu memasukan data *source* dan *destination* kemudian menekan tombol *GetRoute* untuk menampilkan rute navigasi pada peta 3D.
3. Melakukan proses navigasi, dengan kondisi awal pengguna belum bergerak. untuk melakukan proses navigasi ada 3 macam skenario yang dapat dilakukan, menekan dan menggerakan tombol *Rotate* untuk merubah posisi hadap pengguna, menekan tombol *Forward* untuk menggerakan penguna maju ke depan, dan menekan tombol *Backward* untuk menggerakan pengguna mundur ke belakang.

Pengujian kinerja dilakukan langsung ke beberapa *smartphone* dengan cara menggunakan aplikasi untuk melakukan navigasi pada keadaan sebenarnya. Pengujian ini bertujuan untuk mengukur kinerja sistem pada lebih dari satu *smartphone* dengan parameter kecepatan menampilkan peta 3D dan rute perjalanan. Hasil yang didapatkan menunjukan bahwa aplikasi bisa berjalan dengan lancar pada semua *smartphone*.

Analisa algoritma dilakukan dengan membandingkan algoritma *Bi-Directional Dijkstra* sebagai algoritma utama dan algoritma *Dijkstra* sebagai algoritma pembanding. Pada pengujian ini digunakan titik awal dan titik akhir yang sama namun *running time* yang diperoleh berbeda. Iterasi yang dilakukan kedua algoritma tersebut membutuhkan waktu $O(\log n)$, *running time* yang didapatkan *Bi-Directional Dijkstra* sebesar $O(\sqrt{n}\log n)$, sedangkan *Dijkstra* didapatkan *running time* selama $O(n \log n)$. Dengan demikian algoritma *Bi-Directional Dijkstra* memiliki *running time* lebih cepat dibandingkan algoritma *Dijkstra*. Hal ini karena algoritma *Dijkstra* mencari semua kemungkinan rute dengan cara mengecek semua *vertex* pada *graph* dari *source* menuju *destination*, sedangkan algoritma *Bi-Directional Dijkstra* mencari rute dengan menjalankan algoritma *Dijkstra* dari *source* menuju *destination* dan dari *destination* menuju *source* dan akan berhenti ketika kedua *path* bertemu dan digabungkan menjadi satu *path*.

Tabel 1.
Hasil pengujian fungsionalitas

No	Nama Pengujian	Hasil Pengujian
1	Menampilkan posisi pengguna	Berhasil
2	Menampilkan rute navigasi	Berhasil
3	Melakukan proses navigasi	Berhasil

Tabel 2.
Hasil pengujian kinerja

No	Jenis *Smartphone*	Kecepatan Menampilkan	
		Peta 3D	Rute
1	Oppo R7 Lite	6 ms	3 ms
2	Xiaomi Redmi Note 2	6 ms	3 ms
3	Infinix Hot 2	7 ms	5 ms

Tabel 3.
Hasil analisa algoritma

No	Nama Algoritma	*Runtime*
1	*Dijkstra*	$O(n \log n)$
2	*Bi-Directional Dijkstra*	$O(\sqrt{n}\log n)$

Hasil pengujian fungsionalitas ditunjukan oleh Tabel 1, hasil pengujian kinerja ditunjukan oleh Tabel 2 dan hasil analisa algoritma ditunjukan oleh Tabel 3.

VI. KESIMPULAN/RINGKASAN

Selama proses perancangan, implementasi, dan pengujian dapat diambil kesimpulan sebagai berikut:

1. Proses memanfaatkan sistem *Indoor Localization* menggunakan sinyal *Wi-Fi* sebagai penentu posisi pengguna pada sistem *3D Indoor Navigation* dilakukan dengan memanfaatkan fungsi *Clustering Filtered k-Nearest Neighbors* pada sistem *Indoor Localization* dan menerapkannya pada sistem *3D Indoor Navigation*.
2. Penerapan algoritma *Bi-Directional Dijkstra* dalam mencari rute terpendek dilakukan dengan menggunakan *graph* yang terdiri dari *vertex* dan *edge*. *Vertex* menunjukan lokasi ruangan dan persimpangan pada Gedung Teknik Informatika dan *edges* adalah jarak antara dua lokasi yang bersebelahan.
3. Pengguna dapat mengetahui rute perjalanan dengan memasukan data asal pada kolom *source* dan data tujuan pada kolom *destination*. Informasi rute perjalanan ditunjukan dengan garis berwarna ungu pada halaman utama.
4. Algoritma *Bi-Directional Dijkstra* terbukti lebih cepat daripada algoritma *Dijkstra*. Hal ini ditunjukan dengan perbandingan kompleksitas yaitu untuk algoritma *Bi-Directional Dijkstra* $O(\sqrt{n}\log n)$ dan untuk algoritma *Dijkstra* $O(n \log n)$.

UCAPAN TERIMA KASIH

Penulis M.A.M.P. mengucapkan terima kasih kepada Jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember atas segala bantuan dan bimbingannya dalam menempuh ilmu dan meraih gelar Sarjana dalam tahun 2012-2016.

DAFTAR PUSTAKA

[1] H. Fredrick, "Why Doesn't GPS Work Inside a Building?," OpposingViews.com, [Online]. Available: http://science.opposingviews.com/doesnt-gps-work-inside-building-18659.html. [Accessed 25 May 2016].

[2] S. Steiniger, M. Neun and A. Edwardes, "Foundations of Location Based Services," *Lecture Notes on LBS,* 2006.

[3] M. F. Ghanianto, Implementasi Indoor Localization Menggunakan Sinyal Wifidan Clustering Filtered K-Nearest Neighbors untuk Pelacakan Keberadaan Seseorang dan Evaluasi Akurasi Pelacakan di Kampus Teknik Informatika ITS, Surabaya, 2015.

[4] Unity Technologies, "Unity Game Engine," 2016. [Online]. Available: https://unity3d.com/. [Accessed 13 May 2016].

[5] "About SQLite," [Online]. Available: https://www.sqlite.org/about.html. [Accessed 11 June 2016].

[6] M. Sniedovich, "*Dijkstra*'s Algorithm," The University of Melbourne, 27 February 2005. [Online]. Available: http://www.ms.unimelb.edu.au/~moshe@unimelb/620-261/*Dijkstra*/*Dijkstra*.html. [Accessed 7 Juny 2016].

[7] M. Dramski, "*Bi-Directional* search in route planning in navigation," *Scientific-Journals,* pp. 57-62, 2014.